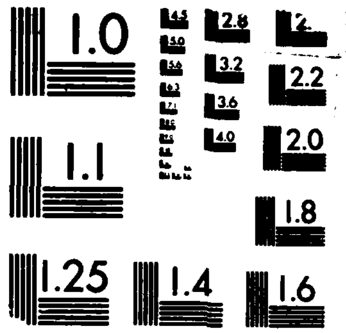
AD-A193 673   PARALLEL REAL-TIME EXPERT SYSTEMS(U) OHIO STATE UNIV   1/1
COLUMBUS B CHANDRASEKARAN ET AL. 20 FEB 88
AFOSR-TR-88-0558 AFOSR-87-0076

UNCLASSIFIED   F/G 12/6   NL



MICROCOPY RESOLUTION TEST CHART
UREAU OF STANDARDS-1963-A

DTIC FILE COPY

AD-A193 673

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

(2)

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY: UNCLASSIFIED	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release, distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S): AFOSR-TR- 88-0558 ~~AFOSR~~
6a. NAME OF PERFORMING ORGANIZATION: Ohio State University; 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION: AFOSR
6c. ADDRESS (City, State and ZIP Code): Columbus Ohio 43212-1194	7b. ADDRESS (City, State and ZIP Code): BLDG #410 Bolling AFB, DC 20332-6448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: AFOSR; 8b. OFFICE SYMBOL (If applicable): NM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: AFOSR- 87-00~~90~~ 76
8c. ADDRESS (City, State and ZIP Code): BLDG #410 Bolling AFB, DC 20332-6448	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO
61102F	2304	A5	

11. TITLE (Include Security Classification): Parallel Real-Time Expert Systems

12. PERSONAL AUTHOR(S): Chandrasekaran

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final	FROM 11/01/86 TO 4/30/88	88/02/20	4

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report summarizes work done under a grant issued through the University Research Instrumentation Program. The grant was issued to support the purchase of parallel processing equipment to conduct experimental research in high-speed real-time decision making by parallelism. The facilities were used ro understand the software requirements of real-time systems, such as complex robotics systems.

[illegible] multiprocessors; CHACS (Concurrent [illegible] Hierarchical Adaptable Object System)

DTIC ELECTE MAY 19 1988 S E D

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED [✓ SAME AS RPT] ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: Abraham Waksman	22b. TELEPHONE NUMBER (Include Area Code): 202-767- 5025; 22c. OFFICE SYMBOL: NM

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

Accession For	
NTIS GRA&I	✓
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

Progress Report on AFOSR Grant 87-0076

University Research Instrumentation Program

Parallel Real-Time Expert Systems

February 20, 1988

B. Chandrasekaran, PI, and Karsten Schwan, Co-PI

AFOSR-TR- 88-0558

1. Introduction

The purpose of this grant was to acquire parallel processing hardware to conduct experimental research in high-speed, real-time decision-making by parallelism, in particular as motivated by problems in AI and robotics. As a result of on-going expansion of such facilities in the Department of Computer and Information Science, we were able to pool the grant resources and other resources so as to acquire a variety of parallel processing hardware for comparative research. The AFOSR funds paid for part of a BBN Butterfly computer facility, while other resources paid for the acquisition of an Intel Hypercube.

DTIC COPY INSPECTED 1

2. Configuration of the BBN Butterfly

Due to the negotiations with BBN for substantially discounted prices, the need for coordination of other parallel processing hardware, and the University purchasing regulations, the hardware was delivered only in late 1987. Additional expansion of the configuration is being planned.

Current Configuration. The system that is currently installed includes the following: The Butterfly currently has 6 68020 nodes with 1Mbyte of memory each and it runs BBN's Chrysalis operating system. The Butterfly C cross compiler and the Fortran compiler are installed on then front end Sun workstation. The Uniform System libraries (native and Sun) are installed on the suns.

February/March Upgrade. The following are the scheduled upgrades for February or March of 1988: Four additional nodes will be installed, and all nodes will be increased to 4 Mbytes of memory. Scheme A Scheme compiler will be installed. And all nodes will be upgraded to Butterfly Plus nodes when they are available.

May/June Upgrade. The last phase of the upgrades will transform the Butterfly Plus into a Butterfly GP1000. This phase will include the following: Disk A disk controller and 500 Mbyte disk will be installed as will a Tape A tape drive. The GP1000 will run a version of CMU's Mach operating system. Common Lisp Common Lisp will be installed.

As mentioned earlier, the AFOSR funds would have paid for a part of the above configuration with substantial cost-sharing by the University.

88 5 16 122

3. Research Progress and Plans

The initial research using the Butterfly was undertaken in the area of operating systems for parallel processing for robotics. The AI research that is being planned, viz., the development of the Generic Task toolset in commonlisp to run on the Butterfly has begun, but no Butterfly implementations have started yet.

Thus a large part of this report will deal with the progress in the research on parallel operating systems for robotics.

3.1. Use of the Butterfly Multiprocessor for Real-Time Robotics

Investigator: Karsten Schwan and associates. During the last few years, we have made significant progress in understanding the software requirements of real-time systems, such as the complex, robot vehicle developed with DARPA funds at the Ohio State University, the ASV robot. Specifically, after having designed and implemented the operating system for the ASV robot's embedded multiprocessor, we developed a follow-up operating system (termed CHAOS) now running on our laboratory replica of the embedded multiprocessor. Since it is apparent that functionality like that offered by CHAOS is necessary for real-time systems[1], two graduate students are now porting CHAOS to the BBN Butterfly multiprocessor, on the base of the Butterfly's current Chrysalis operating system. This port is roughly 30% complete, and should provide us with a rich, experimental base for the development of real-time and robotic applications. Its dependencies on Chrysalis are minimal, thereby making it easy for us to repeat the CHAOS port to the Mach operating system now being developed for the Butterfly by BBN.

The CHAOS basis for real-time program execution should be of use to other groups at OSU, as well. For example, researchers in Electrical Engineering (David Orin) are now cooperating with researchers in Computer and Information Science (P. Sadayappan) to evaluate their ideas regarding the construction of high-performance robotics applications. In addition, since CHAOS is object-based, we are planning to use CHAOS to understand the performance issues in runtime system support for executable, distributed and parallel Ada programs (a proposal to ONR's Real-Time Systems Initiative is now being prepared by us).

3.1.1. The CHAOS Multiprocessor Operating/Runtime System Kernel

The CHAOS (Concurrent Hierarchical Adaptable Object System) was designed and implemented in response to our need for a better methodology of structuring complex, real-time operating software, and also to facilitate the static and dynamic adaptation of such software. CHAOS has been developed in the PArallel Real-Time Systems (PARTS) Laboratory at the Ohio State University. This facility provides a testbed for hierarchical, parallel, real-time software that is a replica and extension of the computer hardware embedded in current, complex, robot vehicles, such as the ASV and ALV vehicles, the first of which was developed with DARPA funding at The Ohio State University.

CHAOS offers primitives that allow high-performance, large-scale, real-time software to be programmed as a system of interacting objects. Its uses the following three principles of software development:

[1]The company (AMT Corporation) carrying out much of the follow-up research for the ASV project and also bidding on the next generation prototype of the ASV is now developing a version of CHAOS that may execute on inhomogeneous, embedded machines, like those consisting of general-purpose processors as well as Lisp machines for high-level planning or vector machines for the processing of vision input.

1. Minimal "hardwired" runtime functionality;
2. Customization by allowing programmers to select among alternative primitives, to tailor primitives by parameterization, and even to synthesize customized primitives exhibiting the precise, desired functionality and performance.

Furthermore, CHAOS demonstrates that the object model of software can be specialized and implemented efficiently so that it may be used in the real-time domain. The following specializations and implementation attributes exist:

- Objects of different weights may be created, ranging from light-weight, passive objects that have no internal processes to heavy-weight objects that may have multiple internal processes. Therefore, an object may exhibit internal parallelism.
- CHAOS objects interact by means of invocations. In order to implement efficient object interactions, multiple CHAOS primitives exist for the invocation of an object's operations. These primitives differ in their semantics, performance, lifetimes, and reliabilities; and their diversity emphasizes the fact that existing implementations of objects or of RPC semantics for computer networks cannot be trivially applied to the real-time domain. The current CHAOS invocation primitives range from (a) *ObjFastInvoke* / fast control invocations that may be used to toggle actions through (b) *ObjInvoke* / invocations that entail the transfer of control and parameter-passing (much like RPC implementations) to (c) *ObjStreamInvoke* / streaming invocations with low incremental cost of data transfer (similar to streaming sockets in Berkeley 4.2 Unix^{TM}). In addition, alternative implementations may be selected based on (a) whether or not the invoker intends to block on the status of the invocation (synchronous vs. asynchronous invocations) and on (b) whether or not all resources acquired for the invocation (eg. parameter blocks) must explicitly be released after each invocation or whether the cost of releasing resources can be amortized over several invocations (persistent invocations). Since invocations may consume memory resources, CHAOS also provides primitives for memory management and garbage collection.
- Explicit scheduling parameters and real-time constraints can be attached to object invocations, and the queueing policy for invocations in the target object can be controlled and changed, as well.

3.1.2. The ARTS Extension of CHAOS

Another important issue in real-time systems being addressed by a Ph.D. thesis in Computer and Information Science is the guarantee of timeliness in the execution of multiple, interacting processes performing a single task. CHAOS-ARTS (Atomic Real-time Transactions Support) is an extension of CHAOS being developed on the BBN Butterfly that supports nested atomic transactions, able to guarantee the timely completion of multiple, possibly parallel actions and providing mechanisms to programmers to recovery from timing errors.

Roughly 40% of a first prototype of ARTS has been implemented to date. This initial version of ARTS supports only 'active', 'heavy-weight' objects, which may have internal parallelism (multiple execution threads). Such objects can be either atomic or non-atomic. Invoking an atomic object starts a new transaction. The new transaction can either be nested in the invoker's hierarchy or can start a new one. An invocation can either commit (succeed) or abort (fail). Concurrency atomicity is implemented using two-phase-locking. Locks are local to objects and can only be obtained by invocations. Failure atomicity is supported by both backward recovery (undoing state changes) and forward recovery (compensating operations).

The ARTS design recognizes the special nature of real-time software. Timing constraints are part of the invocation/transaction mechanism. Both deadlines and delays can be specified for an invocation. Deadlines propagate down the transaction tree. They can be either soft or hard. Soft deadlines are used for scheduling purposes.

In ARTS, the synchronization/timing/event mechanisms are integrated with the transaction mechanism; external events, for example, can automatically start/abort invocations. Periodic activities can be specified by connecting a timer to an invocation.

The contradiction between two phase locking and responsiveness is relaxed by defining revocable and interruptible locks. Revoking a lock aborts any invocation(s) with that lock while interrupting a lock simply delays the invocation(s).

Forward recovery is intended for embedded applications in which the external environment represents an integral part of the system's state.

The 'relatively' high overhead introduced by the transaction mechanism is compensated by selective pre-scheduling. Here, we make use of the 'slack times' inherent in any real-time application by performing 'most' of the transaction management while other activities are delayed.

3.2. The Generic Task Toolset

Investigators: B. Chandrasekaran, J. Josephson, T. Bylander and associates. The AI project will in the initial stages mainly concentrate on the development of a high-level toolset, called the Generic Task Toolset, for the construction of knowledge-based systems. The ideas have been outlined in earlier reports to AFOSR for our grant AFOSR 87-0090 and have been published in the technical literature[2]. We have enclosed as an appendix a recent paper that sets out the Toolset concept in some detail. Before we this toolset is implemented on the Butterfly, we need to implement the toolset in Commonlisp first. We are in the process of acquiring a Symbolics lisp machine as front-end to make this process easier. Currently the Commonlisp implementation is coming along well, with support from our DARPA project and from IBM and DEC. We expect to move to the Butterfly stage of the project next year.

3.3. Other Research

As the hardware gets stabilized a number of other projects will get started, some part of the original proposed research and others new projects that will thrive in the new environment. An example of the latter is the newly initiated project on computer graphics, implementing parallel scanline display algorithms.

[2]For example, B. Chandrasekaran, "Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Design," *IEEE Expert*, 1(3), pp. 23-30

END
DATE
FILMED
7-88
DTIC